MULTIMEDIA UNIVERSITY

STUDENT ID NO

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 1, 2016/2017

DCS5088 – OBJECT ORIENTED PROGRAMMING

(For DIT students only)

22 OCTOBER 2016
2.30 pm – 4.30 pm
(2 Hours)

INSTRUCTIONS TO STUDENTS

1. This examination paper consists of **14** pages. There are **TWO (2) sections** in this paper.
2. **SECTION A:** There are **3** structured questions.
3. **SECTION B:** There is **1** structured question.

SECTION A: Structured Questions (Total: 70 Marks)

Instruction: Please write all your answers in the Answer Booklet provided.

QUESTION 1 (30 marks)

1.1 Write the source codes for a function called ***void get_CakeSize ( )***. Your codes must refer to the following requirements:

- Get positive values from user for height, length and width.
- Calculate the cake size by multiplying all the values.
- If the user keys in negative value(s), the size of cake will not be calculated.
- [**Note:** Refer to the sample outputs given below. The ***bold*** items are the inputs entered by user]

[7 marks]

Sample #1 output screen

Positive Input
Enter Cake's width, length, height (in inches): *12 14 10* The size of the cake ordered: 1680 inch Thank You.

Sample #2 output screen

Negative Input
Enter Cake's width, length, height (in inches): *-11 14 10* You've entered negative values! The cake size can't be calculated.

1.2 For the program below, answer the following questions:

```
#include <iostream>                                          //line1
using namespace std;                                         //line2
int main ()                                                  //line3
{  int trial = 0;                                            //line4
   float TScore = 0;                                         //line5
   float Marks = 0;                                          //line6
                                                             //line7
   cout<<"How many Test(s) you have attempted?";             //line8
   cin>> trial;                                              //line9
                                                             //line10
   for ( i=1; i <= trial; i++)                               //line11
   {  int input;                                             //line12
      Cout<<"\n Your Score for Quiz " << i <<":";            //line13
      cin>> input;                                           //line14
      TScore += input;                                       //line15
                                                             //line16
      cout<<"Actual Marks for Quiz " <<i<<":";               //line17
```

Continued...

```
      cin>> input;                                                    //line18
      Marks += input;                                                 //line19
   }                                                                  //line20
                                                                      //line21
 cout<<"Your Final Score is "<<TScore<<" / "<<Marks<<end;             //line22
 cout<<"The percentage you gained is:"                                //line23
 <<((TScore/Marks)*100)<<"% \n";                                      //line24
                                                                      //line25
 return 0;                                                            //line26
}                                                                     //line27
```

a) Identify **THREE (3)** lines that contain error. Fix the errors by writing down the correct answer. [3 marks]

b) Once the corrections are made, trace the output for this program. Assume the user key-in these details as shown below.

Quiz No of Attempts	Quiz 1 details	Quiz 2 details
2	Score : 20	Score : 10
	Actual Marks : 40	Actual Marks : 30

[7 marks]

1.3 Given the incomplete program shown below, answer the following questions:

```
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;
const double PI = 3.14,
ICEPACK_SIZE = 3;   // in litre

class IceCream
{     double ps, packSizeRad, IcePack, bowlPerPack;
  public:
           void input()
           {  cout<<"\nEnter the Ice-Cream Pack size (in Litre): ";
              cin>>ps;
            }

            void calculate()
            {  packSizeRad = ps / 2;
               IcePack = PI * pow(packSizeRad, 2);
               bowlPerPack = IcePack / ICEPACK_SIZE;
            }

            double getServeBowlPerIce()
            { return bowlPerPack;    }
};

int main()
{     // 1.3 a) Write your answer on your answer booklet
```

Continued...

```
    for(int i=0; i<3; i++)
    { // 1.3 b) Write your answer on your answer booklet
    }

    return 0;
}
```

Sample output screen

```
Enter the Ice-Cream Pack size (in Litre):5
This Ice-Cream pack can be served up to 7 Special bowl(s).

Enter the Ice-Cream Pack size (in Litre):8
This Ice-Cream pack can be served up to 17 Special bowl(s).

Enter the Ice-Cream Pack size (in Litre):15
This Ice-Cream pack can be served up to 59 Special bowl(s).
```

[**Note:** Refer to sample output given above. The ***bold*** items are the inputs entered by user]

a) Declare an array of three object elements for class *IceCream* at segment labelled *'// 1.3 a)'*. [1 mark]

b) Complete the segment labelled *'// 1.3 b)'* to do the following for every array element:

(i) Call *input()*. [1 mark]

(ii) Call *calculate()*. [1 mark]

(iii) Display the number of bowls that can be served by calling function *getServeBowlPerIce()* and display the result in a rounded number. [3 marks]

1.4 Given the incomplete program shown below, answer the following questions:

```
#include <iostream>
using namespace std;
class Photograph
{   private:
      int request;
      float sale;
    public:
      float Payment();

      void setSale (float Sal)
      { sale = Sal; }

      void setRequest(int Req)
      { request = Req; }
};
// 1.4 b) Write your answer on your answer booklet

int main()
```

Continued...

```
{   Photograph Convo;
    Convo.setSale(58.72);
    Convo.setRequest(5);
    cout<<"The total payment is:"<< Convo.Payment() <<endl;
    return 0;
}
```

a) List TWO member functions for this program. [2 marks]

b) Define ***float Payment ( )*** function outside the class as shown in segment labelled *'// 1.4 b)'* . The function will calculate the payment by multiplying *request* with *sale* and return the total.

[3 marks]

c) There is an error when the following statement (highlighted in bold and grey) is added in the **main()** function of the same program. Briefly explain why it is an error. [2 marks]

```
int main()
{
    Photograph Convo;
    Convo.setSale(58.72);
    Convo.setRequest(5);
    cout<<" The special sale given is "<< Convo.sale <<endl;
    cout<<"The total payment is:"<< Convo.Payment()<<endl;
    return 0;
}
```

QUESTION 2 (20 Marks)

2.1 Given the program shown below, answer the following questions:

```
#include<iostream>
using namespace std;
class Product
{ string id, year;
  // 2.1 a) write your answer on your answer booklet
  public:
    void setProduct(string pid, string y)
    { id = pid;
      year = y;
    }
};

class Defect
{  string day[5];
   int qty[5];
   Product JC;

  public:
```

Continued...

```
        void setdata(string d[], int no[])
        {  string pid, y;
           cout<<"Enter the product id        :";
           getline(cin, pid);
           cout<<"Enter the manufactured year :";
           getline(cin, y);

           //2.1 b) write your answer on your answer booklet

        }

        void findminmax()
        {
          string minday, maxday;
          int min, max;

           //2.1 c) write your answer on your answer booklet

          cout<<"\n\nDefect report"<<endl;
          cout<<"--------------"<<endl;
          cout<<"Product id :"<<JC.id<<", year manufactured"
          ":"<<JC.year<<endl;
          cout<<"Min defect :"<< min <<" Day :"<< minday <<endl;
          cout<<"Max defect :"<< max <<" Day :"<< maxday <<endl;;

        }
};

int main()
{
   int i, no[5] = { 3, 7, 0, 5, 2};
   string day[5] = { "Mon", "Tues", "Wed", "Thurs", "Fri"};
   Defect D1;
   D1.setdata(day, no);
   D1.findminmax();
   return 0;
}
```

[**Note:** Refer to sample output given below. The ***bold*** items are the inputs entered by user]

Sample output screen

```
Enter the product id        : C5DD12
Enter the manufactured year : 2015


Defect report
--------------
Product id : C5DD12, year manufactured : 2015
Min defect : 0 Day : Wed
Max defect :7 Day : Tues
```

Output 2.1 c)

a) At segment labelled *'//2.1 a)'*, write the codes to allow *Defect* class to access private data members of *Product* class. [1 mark]

Continued...

b) At segment labelled *'//2.1 b)'*, write the codes to:

- Call *setProduct(....)*, passing the required variables using data member object *JC*. [1 mark]
- Copy the contents of arrays from the main function that's passed to the data members (*day[], qty[]*). [2 marks]

c) At segment labelled *'//2.1 c)'*, write the codes using the variables provided (*minday, maxday, min, max*) in the *findminmax()* function to determine the minimum and maximum values of the *qty* array in order to display the output as seen in Label "***Output 2.1 c***)" in the sample output screen. [8 marks]

2.2 Given the program shown below, answer the following questions:

```
#include<iostream>
#include<iomanip>
using namespace std;
class Human
{ protected:
     string name; char gender;
     int age;
     float weight;

     Human(string n, char g, int a, float w)
     { name = n;
       gender = g;
       age = a;
       weight = w;
     }
};

class Athlete : public Human
{  int heart_rate, time;
   float calories_burned;
   public:

    //2.2 a) write your answer on your answer booklet

     void calculate()
     { if(gender == 'M' || gender == 'm' )
            calories_burned = ((age * 0.2017) - (weight * 0.09036) +
(heart_rate * 0.6309) - 55.0969 ) * time / 4.184;
       else if (gender == 'F' || gender == 'f' )
            calories_burned = ((age * 0.074) - (weight * 0.05741) +
(heart_rate * 0.4472) - 20.4022 ) * time / 4.184;
         else
           { calories_burned = 0;
             cout<<"**invalid gender value**"<<endl;
           }
       }

       void display()
       {  cout<<"Name       : "<<name<<endl;
```

Continued...

```
        cout<<"Age          : "<<age<<endl;
        cout<<"Gender       : "<<gender<<endl;
        cout<<"Weight       : "<<weight<<" lbs "<<endl;
        cout<<"Heart Rate                : "
         <<heart_rate<<" bps "<<endl;
        cout<<"Duration of exercise      : "<<time<<" mins "<<endl;
        cout<<"Calories burned           :"
        <<fixed<<setprecision(2)<<calories_burned<<endl;
      }
};

int main()
{  Athlete A("Rizal", 'M', 32, 173, 133, 37);
   A.calculate();
   A.display();
   return 0;
}
```

Sample output screen

```
Name          : Rizal
Age           : 32
Gender        : M
Weight        : 180 lbs
Heart Rate                  : 133 bps
Duration of exercise        : 37 mins
Calories burned             : 168.04
```

a) At segment labelled *'//2.2 a)'*, write the codes to create a constructor for class *Athlete*:

- Parameterized constructor with 6 parameters to set the data members ***name, gender, age, weight, heart_rate and time.*** Use the proper notation to call *Human* class's parameterized constructor passing the necessary variables. [4 marks]

b) Dynamic memory allocation is not utilized at the main function. Rewrite the main function to incorporate dynamic allocation. [4 marks]

QUESTION 3 (20 Marks)

3.1 Given the following incomplete program, define the member function that overloads the '+' addition operator at the segment labelled *'//3.1'*. Refer to the sample output. [3 marks]

```
#include<iostream>
using namespace std;
```

Continued...

```
class Receipt
{     float total;
     public:
        Receipt(float t)
        { total = t;
        }

        //3.1 write your answer on your answer booklet

         void print()
         { cout<<total<<endl;
         }

};

int main()
{  Receipt a1(500.00) , a2(120.55);
   Receipt a3 = a1.operator+(a2);
   cout<<"Combined total is ";
   a3.print();
   return 0;
}
```

Sample output screen

```
Combined total is 620.55
```

3.2 Observe the program that simulates a lottery using a class *Lottery* with the sample outputs. The first lottery object (*WinningDigits*) will have a randomly generated number in the range of 0 through 9. The second object (*Player)* will have the set of 3 validated numbers (**must be in the range of 0 through 9**) from the user. The program must compare the numbers between the two objects and show appropriate outputs (eg: how many digits that are matched). Based on the descriptions, answer the following questions:

```
#include <iostream>
#include <iomanip>
#include <cstdlib>
#include <ctime>
int numMatches = 0;
using namespace std;

class Lottery
{ int array[3];

  public:
   void getWinningNumber()
   {
      unsigned seed = time(0);
      srand(seed);
```

Continued...

```
    for (int digit = 0; digit < 3; digit++)
    {
      // Generate and store a random number between 0 and 9
        array[digit] = rand() % 10;
    }
 }

int getInput()
{  int digitsOK = 1;

    for (int digit = 0; digit < 3; digit++)
    {
          cin >> array[digit];
          if (array[digit] < 0 || array[digit] > 9)
                digitsOK = 0 ;

    }
    return digitsOK;
 }

  void display()
  {   for (int digit = 0; digit < 3; digit++)
        cout<<   array[digit] <<" ";
      cout<<endl;
  }

  //3.2 a) write your answer on your answer booklet

};


int main()
{  Lottery WinningDigits, Player;
   int numberOK;
   WinningDigits.getWinningNumber();
   cout << "Enter the 3 digits of your lottery number, separated by
   blanks: ";
   numberOK = Player.getInput();

  //3.2 b) write your answer on your answer booklet

    WinningDigits.display();
    Player.display();

    if(Player == WinningDigits)
    {  cout<<"You have "<<numMatches<<" digit(s) that matches the
       winning number!"<<endl;
    }
    else
    { cout<<"You have "<<numMatches<<"...matches. Wish you better luck
       next round. "<<endl;
    }
   return 0;
}
```

Continued...

Sample #1 output screen

```
Enter the 3 digits of your lottery number, separated by
blanks: 22 10 1

Only digits between 0 and 9 are allowed.
Re-enter the 3 digits of your lottery number, separated by
blanks: 1 22 8

Only digits between 0 and 9 are allowed.
Re-enter the 3 digits of your lottery number, separated by
blanks: 1 0 3

1 1 4
1 0 3

You have 1 digit(s) that matches the winning number!
```

3.2b)

3.2b)

Sample #2 output screen

```
Enter the 3 digits of your lottery number, separated by
blanks: 3 2 11

Only digits between 0 and 9 are allowed.
Re-enter the 3 digits of your lottery number, separated by
blanks: 1 2 2

4 0 7
1 2 2

You have 0...matches. Wish you better luck next round.
```

3.2b)

a) At segment labelled *'//3.2 a)'*, write the codes to create an overloaded operator "==" member function:

- Parameter : *L* (reference object of *Lottery*)
- *numMatches* (a global constant variable) will be incremented whenever there is a match between the array elements of the objects that are being compared.
- If there are zero matches, integer value 0 will be returned. Otherwise integer value 1 will be returned to the function call.

[7 marks]

b) At segment labelled *'//3.2 b)'*, write the codes to validate the data entry by user (make use of function call ***getInput()***) whereby if invalid data (any of the 3 digits is out of the range) is detected, the user will be prompted to reenter the data. Refer to label *'3.2b)'* at sample output. [3 marks]

Continued...

3.3 Trace and write the output produced by the program shown below. [7 marks]

```
#include <iostream>
#include <string>
using namespace std;

class HumanBody
{
      protected:
            string name;
            float height, weight;
      public:
            virtual void display()
            {
                  cout << "Name           : " << name << endl;
                  cout << "Height (meter) : " << height << endl;
                  cout << "Weight (kg)    : " << weight << endl;
            }


            virtual ~HumanBody()
            {  cout<<"Destruct base"<<endl;
            }
};

class BMI : public HumanBody
{
      private:
            float bm;
      public:
            BMI(string nm, float wg, float hg)
            {
                  bm = 0.0;
                  name=nm;
                  height=hg;
                  weight=wg;
            }
            float calcBMI()
            {
                  bm = weight / (height * height);
                  return bm;
            }

            void display()
            {
                  cout << "~Name           : " << name << endl;
                  cout << "~Height (meter) : " << height << endl;
                  cout << "~Weight (kg)    : " << weight << endl;
                  cout << "~BMI            : " << calcBMI()<<endl;
            }

            ~BMI()
            {  cout<<"Destruct Derived"<<endl;
            }
};

int main()
{     HumanBody *hm = new BMI("Dory Lee", 45, 1.6);
      hm->display();
```

Continued...

```
    delete hm;
    return 0;
}
```

SECTION B (Total: 30 Marks)

Instruction: Please write all your answers in the Answer Booklet provided.

Write a **complete program** that gets user input for the sales of three premium designer shoes product for the months of January till May. The program will display a summary showing the total and average sales for each shoes product.

[**Note:** Refer to sample output given below. The ***bold items*** are inputs entered by user.]

Declare a constant for the program:

- *SIZE* (int) : Set it to constant value of 5. This value is for the size of the array *month* of *DesignerShoes* class that is used in this program.

Create class called *Shoes:*

- Protected data member:
  - *type* : string
  - *material* : string

- Public member functions:
  - *setShoes(....)* : Contains two string parameters to set the *type* and *material.*
  - *setSales( )* : Set it as a pure virtual function.

Create class called *DesignerShoes* [**derived publicly** from class *Shoes*]:

- Private data members
  - *name* : string
  - *month* : string[SIZE]
  - *sales* : float[SIZE]
  - *total* : float

- Public member functions:

Continued...

- *Parameterized constructor* : - Contains three parameters of string type. One of the parameter is used to set the *name* and the other two parameters will be passed to function *setShoes(...)* to set the *type* and *material* .
  - Set the array *month* with values "Jan", "Feb", "Mac", "Apr","May".
  - Set the *total* to 0.

- *setSales( )* : - Get user input for *sales* array.
  - Accumulate the *sales* array element in *total*.

- *display( )* : - Display the *name*, *type*, *material* and *total*.

- *Average(Designer&)* : - Set this function as a *friend* of the class. Refer to the instructions after this to define the function.

Create a friend function called *Average(DesignerShoes&)*.

- Parameter : Reference object of *DesignerShoes* class.
- Returns the average of the *total* (data member of *DesignerShoes* class).

In ***main( )***:

- Declare a pointer object, ***p*** of *DesignerShoes* class.
  - Use this pointer to create a dynamic object element. Pass the values **"Bedazzled-Stone101", "Evening", "Synthetic"** that will set the object's *name*, *type* and *material*.
  - Call *setSales( )*.
- Declare a pointer object, ***q*** of *DesignerShoes* class.
  - Use this pointer to create a dynamic object element. Pass the values **"Summer", "Casual", "Cotton"** that will set the object's *name*, *type* and *material*.
  - Call *setSales( )*.
- Declare a pointer object, ***r*** of *DesignerShoes* class.

Continued...

- Use this pointer to create a dynamic object element. Pass the values **BeautifulDay", "Casual", "Leather"** that will set the object's *name*, *type* and *material*.
- Call *setSales( )*.

- For each dynamic object (***p, q, r***).
  - *Call display()*
  - Display the average of the sales by calling *Average(....), passing the* dynamic object.
- Deallocate memory for all the dynamic objects.

Sample output screen

```
-------------------------------------------------------------------------
       Sales Data Entry for Bedazzled-Stone101 Evening wear shoes
-------------------------------------------------------------------------

Enter month Jan sales: 5500.60
Enter month Feb sales: 15155.00
Enter month Mac sales: 15000.05
Enter month Apr sales: 10500.00
Enter month May sales: 105500.00

-------------------------------------------------------------------------
       Sales Data Entry for Summer Casual wear shoes
-------------------------------------------------------------------------

Enter month Jan sales: 6800.50
Enter month Feb sales: 15661.10
Enter month Mac sales: 15000.68
Enter month Apr sales: 10500.90
Enter month May sales: 15100.00


-------------------------------------------------------------------------
       Sales Data Entry for BeautifulDay Casual wear shoes
-------------------------------------------------------------------------

Enter month Jan sales: 9000.00
Enter month Feb sales: 15662.10
Enter month Mac sales: 12000.10
Enter month Apr sales: 17005.30
Enter month May sales: 11000.00

-------------------------------------------------------------------------
                Summary Report
-------------------------------------------------------------------------

           Name        Type     Material     Total Sales(RM)      Average(RM)
           ----        ----     --------     ---------------      -----------
Bedazzled-Stone101     Evening  Synthetic        151655.66          30331.13
            Summer     Casual     Cotton         63063.18          12612.64
      BeautifulDay     Casual     Leather        64667.50          12933.50
```

End of Page.